LAPORAN PENELITIAN

Pengembangan Aplikasi Deteksi Tepi Citra Medis menggunakan *Canny Detector*

Disusun oleh:
B. Yudi Dwiandiyanta, S.T., M.T.

Program Studi Teknik Informatika
Fakultas Teknologi Industri
Universitas Atma Jaya Yogyakarta
2012

LAPORAN PENELITIAN

Pengembangan Aplikasi Deteksi Tepi Citra Medis menggunakan *Canny Detector*

Disusun oleh:
B. Yudi Dwiandiyanta, S.T., M.T.

Program Studi Teknik Informatika
Fakultas Teknologi Industri
Universitas Atma Jaya Yogyakarta
2012

LEMBAR PENGESAHAN LAPORAN PENELITIAN

No. Proposal:

1.	a.	Judul Penelitian	:	Pengembangan Aplikasi Deteksi Tepi Citra Medis menggunakan *Canny Detector*
	b.	Macam penelitian	:	Laboratorium / Lapangan
2.	Personalia Ketua Penelitian			
	a.	Nama	:	B. Yudi Dwandiyanta, S.T., M.T.
	b.	Jenis Kelamin	:	Laki-laki
	c.	Usia saat pengajuan proposal	:	35 tahun 4 bulan
	d.	Jabatan akademik/Golongan	:	Lektor Kepala / IIID
	e.	Fakultas/Prodi	:	Fakultas Teknologi Industri / Teknik Informatika
3.		Lokasi Penelitian	:	Laboratorium Komputasi FTI UAJY
4.		Jangka waktu penelitian	:	6 bulan
5.		Biaya yang diperlukan	:	Rp. 3.030.000,00

Yogyakarta, 20 April 2012
Ketua Peneliti,

B. Yudi Dwiandiyanta, S.T., M.T.

Dekan Fakultas Teknologi Industri,

Ir. B. Kristyanto, M.Eng., Ph.D.

Ketua LPPM,

Dr. Ir. Djarot Purbadi, M.T.

INTISARI

Penggunaan pengolahan citra digital berkembang pesat sejalan dengan berkembangnya teknologi komputer di segala bidang. Beberapa contoh bidang kehidupan yang membutuhkan pengolahan citra digital di antaranya adalah bidang kesehatan: segmentasi untuk membedakan bagian-bagian sel darah, deteksi kerusakan organ tubuh, deteksi keberadaan tumor; bidang teknologi industri: deteksi tepi untuk pengenalan pola hasil produksi dengan membedakannya dengan *background*; bidang geografi: *filtering* untuk menghilangkan *noise* pada citra pemetaan geografis, klasifikasi dalam pemetaan geografis; dan bidang-bidang lainnya. Deteksi tepi perlu dilakukan karena adanya teori bahwa sistem penglihatan manusia (*Human Visual System* / HVS) menunjukkan beberapa urutan dari deteksi tepi terlebih dahulu sebelum pengenalan warna atau intensitas citra (McCane, 2001).

Dalam penelitian ini akan digunakan deteksi tepi citra dengan menggunakan *canny detector*. Sebagai objek dalam penelitian ini digunakan citra medis dengan format .bmp maupun .jpg dan kedalaman warna 24-bit. Hasil keluaran perangkat lunak yang dikembangkan adalah citra hasil deteksi tepi dengan format .png dan format citra biner. Pengembangan perangkat lunak yang digunakan dalam penelitian ini adalah *software* Matlab 6.1.

Berdasarkan dengan penelitian yang telah dilakukan, aplikasi deteksi tepi citra medis menggunakan canny detector ini telah dapat dikembangkan. Hasil operasi deteksi tepi citra yang dikembangkan akan mengalami gangguan yang signifikan apabila diberikan gangguan *noise salt and pepper*, *histogram equalization*, dan operasi penapisan dengan tapis lolos atas (*High Pass Filtering*). Algoritma yang dikembangkan cukup dapat bertahan terhadap pengolahan citra pemberian *noise Gaussian* dan penapisan lolos bawah (*Low Pass Filtering*).

Keyword: Deteksi tepi, citra medis, *Canny Detector*

KATA PENGANTAR

Puji syukur kepada Allah Yang Kudus, atas berkat dan kasih sayang-Nya, akhirnya penulis dapat menyelesaikan penelitian dengan judul "Pengembangan Aplikasi Deteksi Tepi Citra Medis menggunakan *Canny Detector*" untuk diajukan sebagai penelitian di Lembaga Penelitian dan Pengabdian pada Masyarakat Universitas Atma Jaya Yogyakarta.

Dalam kesempatan ini penulis mengucapkan terimakasih kepada:

1. Dr. Ir. Y. Djarot Purbadi, M.T., selaku Ketua Lembaga Penelitian dan Pengabdian pada Masyarakat Universitas Atma Jaya Yogyakarta.
2. Ir. B. Kristyanto, M.Eng., Ph.D., selaku Dekan Fakultas Teknologi Industri Universitas Atma Jaya Yogyakarta.
3. Rekan-rekan di Fakultas Teknologi Industri UAJY yang tidak dapat disebutkan satu persatu.

Tak lupa penulis mohon masukan yang bersifat korektif agar tulisan ini dapat menjadi lebih baik. Akhir kata, semoga tulisan ini dapat bermanfaat bagi kita semua.

Yogyakarta, April 2012

Penulis

DAFTAR ISI

BAB I
PENDAHULUAN

Penggunaan pengolahan citra digital berkembang pesat sejalan dengan berkembangnya teknologi komputer di segala bidang. Beberapa contoh bidang kehidupan yang membutuhkan pengolahan citra digital di antaranya adalah bidang kesehatan: segmentasi untuk membedakan bagian-bagian sel darah, deteksi kerusakan organ tubuh, deteksi keberadaan tumor; bidang teknologi industri: deteksi tepi untuk pengenalan pola hasil produksi dengan membedakannya dengan *background*; bidang geografi: *filtering* untuk menghilangkan *noise* pada citra pemetaan geografis, klasifikasi dalam pemetaan geografis; dan bidang-bidang lainnya.

Pengolahan citra digital digunakan untuk melakukan interpretasi terhadap citra digital. Beberapa teknik yang digunakan dalam pengolahan citra adalah *filtering*, *enhancement*, deteksi tepi, segmentasi, klasifikasi, kompresi, rekonstruksi citra dan lain-lain.

Beberapa alasan yang mendukung kegunaan deteksi tepi dalam aplikasi kehidupan sehari-hari adalah :

- Manusia memiliki kecenderungan dalam mengenal suatu obyek atau kecenderungan kumpulan obyek dengan melihat tepi dari citra.
- Adanya teori (yang dapat dijadikan sebagai alasan psikologis), yaitu bahwa sistem penglihatan manusia (*Human Visual System* / HVS) menunjukkan beberapa urutan dari deteksi tepi terlebih dahulu sebelum pengenalan warna atau intensitas citra (McCane, 2001).

Adapun latar belakang dilakukan pembandingan beberapa algoritma deteksi tepi adalah untuk memperoleh hasil tepi yang baik dan dengan waktu yang seefisien mungkin. Kriteria hasil tepi yang baik adalah tidak menyertakan atau setidaknya mengurangi *noise* tanpa harus kehilangan informasi sinyal utama citra, peka terhadap sinyal tepi yang lemah.

BAB II
TINJAUAN PUSTAKA

Deteksi tepi citra yang digunakan untuk citra medis telah banyak dikembangkan. Thangam et. al. (2009) meneliti tentang deteksi tepi citra medis menggunakan diskriminasi tekstur. Zeng et. al. (2008) menggunakan fuzzy-set untuk melakukan deteksi tepi citra medis. Deteksi tepi cira medis dapat juga dilakukan dengan menggunakan pemrograman dinamis untuk menghasilkan deteksi tepi citra yang optimal (Lee et. al., 2001). Deteksi tepi citra dapat pula diterapkan pada runtun data medis (Bingrong et. al., 2008), medical Thermogram (Selvarasu et. al., 2007), citra berderau (Suzuki, 2003; Gonzalez, 2009), deteksi medical x-Ray (Benjelloun et. Al., 2007), dan citra organ 3-D (Naef, et. al. 1996).

Tepi adalah sekumpulan piksel yang terhubung *(connected pixel)* yang berada pada suatu batas antara dua daerah (Thongsongkrit, 2002). Tepi dapat dideteksi dengan melakukan konvolusi menggunakan matriks-matriks yang diperoleh dari derivatif piksel tetangga lokal. Implementasi derivatif yang biasa digunakan untuk deteksi tepi adalah:

a. Deteksi lokal maksima dari derivatif pertama (Operator Gradien).
b. Deteksi *zero-crossing* dari derivatif kedua (Operator Laplacian).

Beberapa operator deteksi tepi yang diimplementasikan dan dianalisis dalam penelitian ini adalah:

1. Operator gradien: Sobel, Prewitt, Isotropik, Schotastic.
2. Operator kompas: Kompas, Kirsch, Robinson.
3. Operator Laplacian.
4. Detektor Canny.

Kriteria hasil detektor tepi yang baik yaitu, kebal terhadap *noise*, memberikan *detail* tepi yang baik, peka terhadap sinyal tepi untuk citra dengan kontras yang kurang baik dan mempunyai beban komputasi yang lebih kecil.

2.1. Operator Gradien

Proses penggunaan operator gradien dengan menggunakan derivatif pertama untuk menemukan tepi dapat dilakukan dengan langkah-langkah:

1. Penentuan gradien citra untuk mengetahui intensitas variasi lokal dengan melakukan konvolusi dengan matriks konvolusi Gx dan Gy. Matriks konvolusi Gx dan Gy diperoleh dari pendekatan diskret derivatif parsial fungsi f(x,y). Penentuan matriks konvolusi ditunjukkan dalam hubungan-hubungan dari persamaan-persamaan berikut.

$$\nabla f(x,y) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} \quad (2.1)$$

Gradien Gx diperoleh dari pendekatan diferensial horisontal atau derivatif parsial terhadap x pada fungsi f(x,y):

$$\frac{\partial f(x,y)}{\partial x} = f(x,y) - f(x-1,y) \quad (2.2)$$

sehingga diperoleh matriks konvolusi

G_x=[1 -1]

Gradien Gy diperoleh dari pendekatan diferensial vertikal atau derivatif parsial terhadap y pada fungsi f(x,y):

$$\frac{\partial f(x,y)}{\partial y} = f(x,y) - f(x,y-1) \quad (2.3)$$

sehingga diperoleh matriks konvolusi

$$G_y = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad (2.4)$$

Dengan langkah yang sama maka dapat ditentukan matriks konvolusi Gx dan Gy dengan ukuran yang berbeda, misalnya 2 x 2, 3 x 3, 5 x 5 dan lain seterusnya.

2. Penentuan *magnitude* citra sebagai tepi:

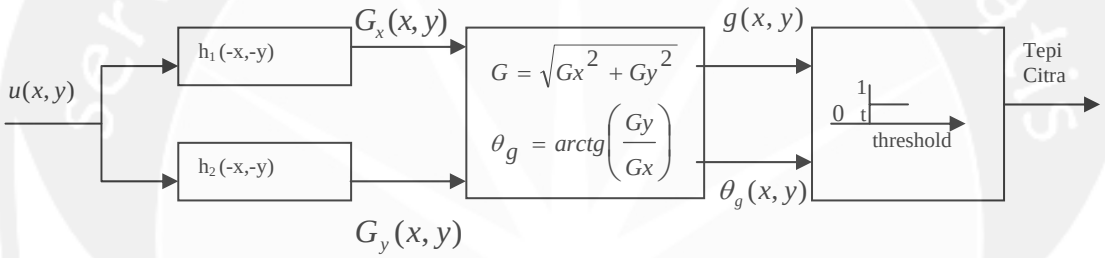
$$magnitude(\nabla f) = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} = \sqrt{G_x{}^2 + G_y{}^2} \qquad (2.5)$$

Penentuan besar sudut atau arah untuk mengetahui kecenderungan arah tepi lokal

$$arah(\nabla f) = \tan^{-1}(G_y / G_x)$$

$$\theta(x, y) = a \tan\left(\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}\right) \qquad (2.6)$$

Secara ringkas, penentuan tepi dengan operator gradien dapat dilihat pada Gambar 2.1 (Jain, 1995).



Gambar 2.1. Diagram Blok Deteksi Tepi Dengan Operator Gradien.

Matriks konvolusi untuk operator gradien yang sering digunakan (Jain, 1995):

1. Detektor Prewitt

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{ dan } G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \qquad (2.7)$$

2. Detektor Sobel

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ dan } G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad (2.8)$$

3. Detektor isotropik

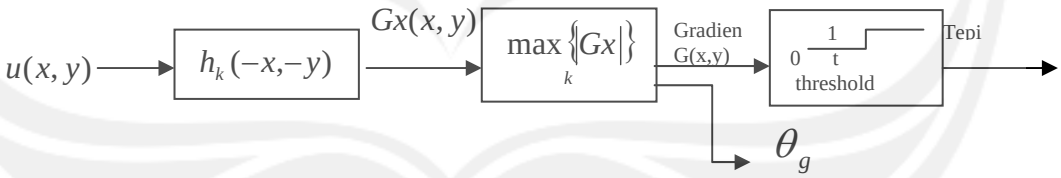
$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix} \text{ dan } G_y = \begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix} \quad (2.9)$$

4. Detektor Stochastic

$$G_x = \begin{bmatrix} -0.776 & 0 & 0.7761 \\ -1 & 0 & 1 \\ -0.776 & 0 & 0.776 \end{bmatrix} \text{ dan } G_y = \begin{bmatrix} 0.776 & 1 & 0.776 \\ 0 & 0 & 0 \\ -0.776 & -1 & -0.776 \end{bmatrix} \quad (2.10)$$

2.2. Operator Kompas

Deteksi tepi menggunakan operator kompas dilakukan dengan menghitung gradien pada 8 arah mata angin kemudian tepi ditentukan dari gradien maksimum (Jain, 1995). Gambar 2.2 merupakan alur operator kompas dalam menentukan deteksi tepi (Jain, 1995).



Gambar 2.2. Diagram Blok Deteksi Tepi Dengan Operator Kompas.

Beberapa operator yang menggunakan algoritma operator kompas yang sering digunakan (Jain, 1995):

1. Matriks konvolusi untuk operator kompas

$$S = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad W = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

$$N=\begin{bmatrix}1 & 1 & 1\\ 0 & 0 & 0\\ -1 & -1 & -1\end{bmatrix} \qquad E=\begin{bmatrix}-1 & 0 & 1\\ -1 & 0 & 1\\ -1 & 0 & 1\end{bmatrix}$$

$$SE=\begin{bmatrix}-1 & -1 & 0\\ -1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix} \qquad SW=\begin{bmatrix}0 & -1 & -1\\ 1 & 0 & -1\\ 1 & 1 & 0\end{bmatrix}$$

$$NW=\begin{bmatrix}1 & 1 & 0\\ 1 & 0 & -1\\ 0 & -1 & -1\end{bmatrix} \qquad NE=\begin{bmatrix}0 & 1 & 1\\ -1 & 0 & 1\\ -1 & -1 & 0\end{bmatrix}$$

Gambar 2.3. Matriks Konvolusi Operator Kompas.

2. Matriks konvolusi untuk detektor Kirsch

$$S=\begin{bmatrix}5 & 5 & 5\\ -3 & 0 & -3\\ -3 & -3 & -3\end{bmatrix} \qquad W=\begin{bmatrix}-3 & 5 & 5\\ -3 & 0 & 5\\ -3 & -3 & -3\end{bmatrix}$$

$$N=\begin{bmatrix}-3 & -3 & 5\\ -3 & 0 & 5\\ -3 & -3 & 5\end{bmatrix} \qquad E=\begin{bmatrix}5 & 5 & -3\\ 5 & 0 & -3\\ -3 & -3 & -3\end{bmatrix}$$

$$SE=\begin{bmatrix}-3 & -3 & -3\\ -3 & 0 & -3\\ 5 & 5 & 5\end{bmatrix} \qquad SW=\begin{bmatrix}-3 & -3 & -3\\ 5 & 0 & -3\\ 5 & 5 & -3\end{bmatrix}$$

$$NW=\begin{bmatrix}5 & -3 & -3\\ 5 & 0 & -3\\ 5 & -3 & -3\end{bmatrix} \qquad NE=\begin{bmatrix}-3 & -3 & 5\\ -3 & 0 & 5\\ -3 & 5 & 5\end{bmatrix}$$

Gambar 2.4. Matriks Konvolusi Operator Kirsch.

3. Matriks konvolusi untuk detektor Robinson

$$S=\begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix} \qquad W=\begin{bmatrix}0 & 1 & 2\\ -1 & 0 & 1\\ -2 & -1 & 0\end{bmatrix}$$

$$N=\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix} \qquad E=\begin{bmatrix}2 & 1 & 0\\ 1 & 0 & -1\\ 0 & -1 & -2\end{bmatrix}$$

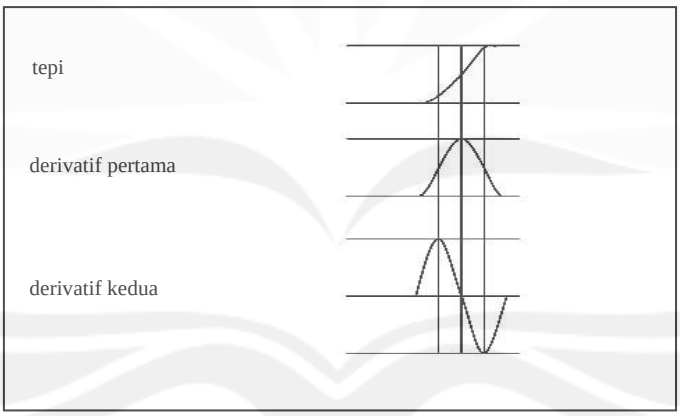
$$SE = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad SW = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$$

$$NW = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad NE = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Gambar 2.5. Matriks Konvolusi Operator Robinson.

2.3. Operator Laplacian

Metode lain untuk menentukan tepi adalah menggunakan derivatif orde kedua. Perbedaan penggunaan derivatif pertama dan kedua dapat dilihat pada grafik berikut (Yaniv, 2002):



Gambar 2.6. Perbedaan Tepi dengan Derivatif Pertama dan Tepi dengan Derivatif Kedua.

Dari Gambar 2.6. maka dapat dilihat terdapat *zero-crossing* antara gradien positif dan negatif yang dihasilkan tiap piksel. Sehingga dapat disimpulkan derivatif kedua bersifat lebih peka terhadap perubahan intensitas piksel. Kepekaan operator derivatif kedua terhadap perubahan intensitas piksel dapat ditunjukkan

dengan kepekaannya terhadap *noise*. Maka dari itu, deteksi tepi dengan Laplacian selalu menggunakan *Gaussian smoothing* untuk mengurangi *noise* terlebih dahulu sebelum dilakukan konvolusi dengan matriks konvolusi.

Operator Laplacian dinotasikan dalam persamaan berikut:

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (2.11)$$

Penurunan salah satu pendekatan matriks konvolusi untuk operator Laplacian ditunjukkan dalam hubungan persamaan-persamaan berikut:

$$\frac{\partial^2 f}{\partial x^2} = f(x-1,y) - 2(f(x,y) + f(x+1,y) \quad (2.12)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x,y-1) - 2(f(x,y) + f(x,y+1) \quad (2.13)$$

$$\nabla^2 f = -4f(x,y) + f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) \quad (2.14)$$

Dari persamaan di atas dan menggunakan acuan matriks D pada Gambar 2.5, maka dapat ditentukan matriks konvolusi operator Laplacian:

$$\nabla^2 f = -4f(x,y) + f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) \quad (2.15)$$

$$\nabla^2 f = -4w5 + (w2 + 24 + w6 + w8) \quad (2.16)$$

Sehingga diperoleh matriks konvolusi Laplacian :

$$G = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (2.17)$$

Matriks konvolusi operator Laplacian yang biasa digunakan adalah (Jain, 1995):

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

(a) (b) (c)

Gambar 2.7 Matriks Konvolusi Deteksi Tepi dengan Operator Laplacian.

2.4. Detektor Tepi Canny

Salah satu metode pengembangan deteksi tepi tradisional yang sering digunakan adalah detektor Canny. Canny memberikan metode penemuan tepi dengan langkah-langkah (Lee, 2002):

1. Melakukan konvolusi dengan matriks Gx dan Gy untuk menentukan gradien citra dengan arah sumbu x dan arah sumbu y, dimana matriks Gx dan Gy ditentukan dari derivatif parsial persamaan Gaussian. Penggunaan derivatif parsial Gaussian untuk menentukan gradien Gx dan gradien Gy dapat dilihat pada hubungan persamaan-persamaan berikut. Persamaan Gaussian $h(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$ memberikan derivatif parsial $Gx = \frac{\partial h}{\partial x} = -\frac{x}{\sigma^2} G$ dan $Gy = \frac{\partial h}{\partial y} = -\frac{y}{\sigma^2} G$ untuk menentukan gradien Gx dan Gy. Langkah selanjutnya adalah menentukan *magnitude* citra dengan rumus $magnitude(\nabla f) == \sqrt{G_x{}^2 + G_y{}^2}$. Persamaan Gaussian jika diimplementasikan sebagai matrik dalam jendela 3 x 3 akan untuk memperoleh matriks Gx dan Gy dapat dilihat sebagai berikut (Yaniv, 2002):

$$h(x,y) = \begin{bmatrix} e^{-\frac{2}{2\sigma^2}} & e^{-\frac{1}{2\sigma^2}} & e^{-\frac{2}{2\sigma^2}} \\ e^{-\frac{1}{2\sigma^2}} & e^{0} & e^{-\frac{1}{2\sigma^2}} \\ e^{-\frac{2}{2\sigma^2}} & e^{-\frac{1}{2\sigma^2}} & e^{-\frac{2}{2\sigma^2}} \end{bmatrix} \quad (2.18)$$

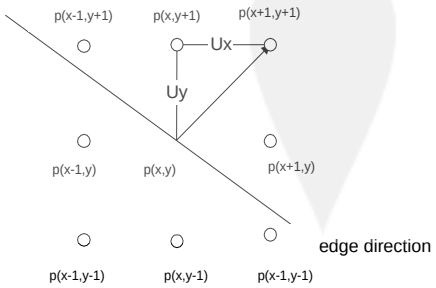
$$Gy = \begin{bmatrix} -\frac{-1}{\sigma^2}e^{-\frac{2}{2\sigma^2}} & -\frac{-1}{\sigma^2}e^{-\frac{1}{2\sigma^2}} & -\frac{1}{\sigma^2}e^{-\frac{2}{2\sigma^2}} \\ 0 & 0 & 0 \\ -\frac{1}{\sigma^2}e^{-\frac{2}{2\sigma^2}} & -\frac{1}{\sigma^2}e^{-\frac{1}{2\sigma^2}} & -\frac{1}{\sigma^2}e^{-\frac{2}{2\sigma^2}} \end{bmatrix} \quad Gx = \begin{bmatrix} -\frac{-1}{\sigma^2}e^{-\frac{2}{2\sigma^2}} & 0 & -\frac{1}{\sigma^2}e^{-\frac{2}{2\sigma^2}} \\ -\frac{-1}{\sigma^2}e^{-\frac{1}{2\sigma^2}} & 0 & -\frac{1}{\sigma^2}e^{-\frac{1}{2\sigma^2}} \\ -\frac{-1}{\sigma^2}e^{-\frac{2}{2\sigma^2}} & 0 & -\frac{1}{\sigma^2}e^{-\frac{2}{2\sigma^2}} \end{bmatrix} \quad (2.19)$$

Sehingga diperoleh matriks konvolusi Gx dan Gy:

$$Gy = \begin{bmatrix} \frac{1}{\sigma^2}e^{-\frac{2}{2\sigma^2}} & \frac{1}{\sigma^2}e^{-\frac{1}{2\sigma^2}} & \frac{1}{\sigma^2}e^{-\frac{2}{2\sigma^2}} \\ 0 & 0 & 0 \\ -\frac{1}{\sigma^2}e^{-\frac{2}{2\sigma^2}} & -\frac{1}{\sigma^2}e^{-\frac{1}{2\sigma^2}} & -\frac{1}{\sigma^2}e^{-\frac{2}{2\sigma^2}} \end{bmatrix} \quad Gx = \begin{bmatrix} \frac{1}{\sigma^2}e^{-\frac{2}{2\sigma^2}} & 0 & -\frac{1}{\sigma^2}e^{-\frac{2}{2\sigma^2}} \\ \frac{1}{\sigma^2}e^{-\frac{1}{2\sigma^2}} & 0 & -\frac{1}{\sigma^2}e^{-\frac{1}{2\sigma^2}} \\ \frac{1}{\sigma^2}e^{-\frac{2}{2\sigma^2}} & 0 & -\frac{1}{\sigma^2}e^{-\frac{2}{2\sigma^2}} \end{bmatrix}$$

Gambar 2.8 Matriks Konvolusi Operator Canny.

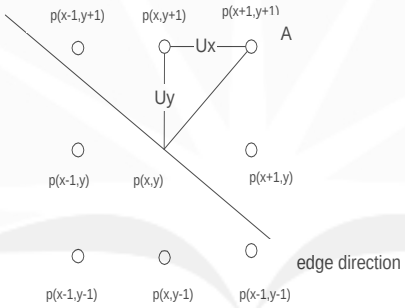
2. Melakukan *non maxima suppression* untuk menipiskan tepi citra yang telah diperoleh pada langkah pertama. Pada langkah ini tiap piksel dilakukan pengecekan apakah piksel tersebut merupakan lokal maksima pada arah gradien, jika ya maka piksel dipertahankan jika tidak maka piksel dihapus. Penentuan lokal maksima pada arah gradien dapat dilakukan menggunakan interpolasi atau ekstrapolasi titik p(x,y) (Lee, 2002).



Gambar 2.9 Interpolasi titik p(x,y).

p(x,y) adalah titik piksel yang dihitung. Ux adalah normal arah gradien dan Uy adalah proyeksi normal terhadap sumbu y. Adapun urutan langkah *non maksima suppresion* adalah:

a) Pada tiap piksel p(x,y) tentukan gradien Gx dan Gy, bandingkan dengan gradien tetangga yang searah, jika gradien tersebut lebih besar dari tetangga yang searah tersebut maka piksel tersebut diberi tanda untuk dipertahankan sebagai piksel tepi, tetapi jika piksel tersebut lebih kecil dari piksel tetangga searah maka dihapus.
b) Melakukan estimasi gradien dengan cara memilih dua piksel terdekat (Lee, 2002).

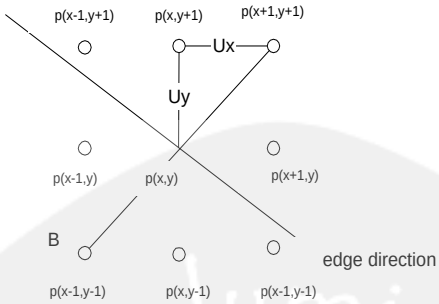


Gambar 2.10 Aproksimasi Magnitudo Gradien Pada Titik A.

Magnitudo gradien pada 3 titik yang membentuk segitiga digunakan untuk menentukan aproksimasi nilai magnitudo gradien pada titik A.

$$G_A = \frac{u_x}{u_y} G(x+1, y+1) + \frac{u_y - u_x}{u_y} G(x, y+1) \qquad (2.20)$$

Gradien terinterpolasi pada sisi B ditunjukkan pada Gambar 11 (Lee, 2002)



Gambar 2.11 Aproksimasi Magnitudo
Gradien Pada Titik B.

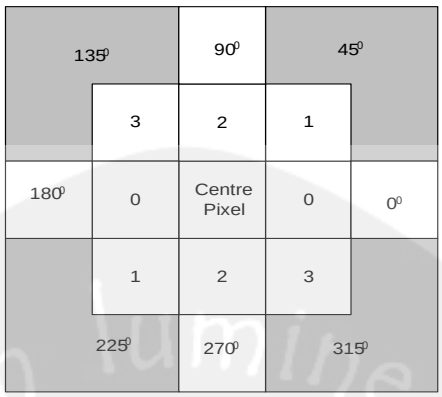
Aproksimasi magnitudo gradien pada titik B ditunjukkan pada persamaan:

$$G_B = \frac{u_x}{u_y} G(x-1, y-1) + \frac{u_y - u_x}{u_y} G(x, y-1) \qquad (2.21)$$

c). Menentukan p(x,y) adalah maksimum jika G(x,y)>GA dan G(x,y)> GB.

Non maxima suppression dapat juga dilakukan dengan cara sebagai berikut:

a) Melakukan klasifikasi arah sudut (θ) menjadi 8 sektor sesuai arah mata angin seperti terlihat pada Gambar 12.
b) Melakukan pengecekan apakah piksel merupakan gradien maksimum pada arah yang sudah ditentukan. Jika piksel bernilai maksimum pada arah gradien maka piksel dianggap sebagai tepi. Jika tidak maka piksel tersebut bukan tepi.



Gambar 2.12

Klasifikasi sudut.

3. Menggunakan *double-thresholding* T1 dan T2 dimana T1>T2.

   a. *Thresholding* pertama

   Semua piksel dengan nilai *magnitude* gradien lebih besar daripada T1 diklasifikasikan elemen tepi.

   b. *Thresholding* kedua

   Dari T2<T1, ambil semua piksel yang lebih besar dari T2 dan jika *magnitude* gradien lebih besar dari T2 maka klasifikasikan sebagai kandidat tepi.

   c. Pada piksel kandidat tepi, tentukan apakah piksel itu tepi atau bukan dengan jika piksel tetangga dalam 8-ketetanggaan adalah tepi maka piksel tersebut adalah elemen tepi.

BAB III
MASALAH, TUJUAN DAN MANFAAT PENELITIAN

3.1. PERUMUSAN MASALAH

Dalam penelitian ini dapat dijabarkan beberapa perumusan masalah yang ada, yaitu :

a. Bagaimana mengembangkan aplikasi deteksi tepi citra medis menggunakan *canny detector*
b. Bagaimana pengaruh noise dan pengolahan citra terhadap algoritma deteksi tepi yang dikembangkan

3.2. TUJUAN PENELITIAN

Tujuan penelitian ini adalah seperti berikut.

a. Mengembangkan aplikasi deteksi tepi citra medis menggunakan *canny detector*
b. Menganalisis pengaruh noise dan pengolahan citra terhadap algoritma deteksi tepi yang dikembangkan.

3.3. MANFAAT PENELITIAN

Kegunaan aplikasi deteksi tepi citra medis menggunakan *canny detector* adalah sebagai berikut :

a. **Bagi pengguna**: menghasilkan tepi citra yang sering digunakan untuk mempermudah analisis data citra.
b. **Bagi peneliti**: mampu mengembangkan dan menerapkan ilmu pengetahuan yang dikuasai.

BAB IV
METODE PENELITIAN

Penelitian ini dilakukan dalam beberapa tahap, sebagai berikut :

4.1. Pengumpulan Bahan

Pengumpulan bahan dengan tujuan untuk memperoleh dasar ilmu yang baik pada penerapan penelitian. Pengumpulan bagan dilakukan dengan mencari buku, jurnal, tesis yang berhubungan dengan penelitian. Pengumpulan bahan dapat memanfaatkan perpustakaan yang ada ataupun mengakses situs-situs internet yang mempublikasikan mengenai penelitian terkait. Berdasarkan bahan-bahan yang diperoleh tersebut kemudian dilakukan pembangunan perangkat keras dan perangkat lunak.

4.2. Perancangan Perangkat Lunak

Tahap ini melakukan penyusunan pemodelan perangkat lunak berdasarkan proses bisnis yang telah dianalisis. Pemodelan dilakukan untuk memudahkan dalam penyusunan perangkat lunak menggunakan *Data Flow Diagram* (DFD)

4.3. Pembuatan Perangkat Lunak

Hasil rancangan pemodelan kemudian diimplementasikan dengan menggunakan Matlab 6.1.

4.4. Pengujian Perangkat Lunak

Perangkat lunak yang telah selesai dibangun kemudian diuji dengan beberapa jenis citra medis. Revisi perangkat lunak dapat dilakukan jika program tidak bekerja sesuai dengan yang diinginkan.

BAB V
ANALISIS DAN PERANCANGAN SISTEM

5.1. Pengantar

Pada bab ini akan dibahas mengenai analisis dan perancangan sistem yang akan dibuat. Pokok bahasan yang terdapat dalam bab ini adalah deskripsi keseluruhan, kebutuhan khusus, kebutuhan fungsionalitas dan perancangan arsitektur sistem yang dikembangkan.

5.2. Deskripsi Keseluruhan

5.2.1. Perspektif Produk

Sistem ini adalah suatu program aplikasi yang digunakan untuk melakukan deteksi tepi citra medis dengan menggunakan *Canny Detector*. Dengan aplikasi ini diharapkan dapat membantu user untuk melakukan deteksi tepi citra medis dan akhirnya membantu user untuk melakukan analisis terhadap citra medis tersebut.

Pada sistem ini, input data yang dapat dimasukkan user adalah: data_citra dan parameter_threshold. Berikut ini adalah proses yang terjadi bila digambarkan dalam sebuah diagram (Gambar 5.1).

Gambar 5.1. Proses pada sistem *standalone*

Data yang terdapat dalam aplikasi ini adalah data citra, yang berupa citra medis dan citra hasil deteksi tepi. Sedangkan *Personal Computer* digunakan untuk menjalankan aplikasi deteksi tepi citra ini.

Pada aplikasi ini, terdapat seorang user yang dapat menggunakan sistem ini. User akan berinteraksi dengan sistem untuk melakukan proses deteksi tepi citra dan menghitung MSE (*Mean Square Error*).

5.2.2. Fungsi Produk

Fungsi produk perangkat lunak yang dikembangkan adalah sebagai berikut:

1. Fungsi deteksi tepi citra, adalah fungsi yang digunakan untuk melakukan deteksi tepi citra medis yang sudah dipilih oleh *user*.
2. Fungsi hitung MSE, adalah fungsi yang digunakan untuk menghitung MSE antara citra hasil deteksi tepi sebelum diberikan gangguan dan sesudah diberikan gangguan.
3. Fungsi operasi pengolahan citra, adalah fungsi yang digunakan untuk memberikan efek pengolahan citra kepada citra.

Selain fungsi-fungsi utama di atas, diberikan juga fungsi-fungsi tambahan sebagai berikut:

1. Fungsi Open, adalah fungsi yang digunakan untuk memilih citra medis, sekaligus menampilkan citra yang dipilih pada sebuah jendela.
2. Fungsi Save, adalah fungsi yang digunakan untuk menyimpan citra hasil deteksi tepi, atau dapat juga digunakan untuk menyimpan citra yang telah diolah dengan beberapa jenis operasi pengolahan citra.

5.2.3. Karakteristik Pengguna

Karakteristik pengguna yang menggunakan perangkat-lunak ini adalah:

a. Mengerti pengoperasian komputer.
b. Memahami sistem komputer tempat perangkat-lunak dijalankan.
c. Mengerti sistem deteksi tepi citra dengan *Canny Detector*.

5.2.4. Batasan-batasan

Sistem ini memiliki keterbatasan, yaitu bersifat *offline*/*standalone*.

5.3. Kebutuhan Khusus

5.3.1. Kebutuhan Antarmuka Eksternal

Kebutuhan antarmuka eksternal pada perangkat-lunak ini meliputi kebutuhan antarmuka pemakai, antarmuka perangkat-keras, antarmuka perangkat-lunak, dan antarmuka komunikasi.

5.3.2. Kebutuhan Antarmuka Internal

Pengguna berinteraksi dengan antarmuka yang ditampilkan dalam layar komputer dengan format windows form dengan pilihan fungsi dan form untuk pengisian data dan tampilan informasi pada layar monitor.

5.3.3. Kebutuhan Antarmuka Perangkat Keras

Antarmuka perangkat keras yang digunakan dalam perangkat-lunak ini adalah:

a. Personal Komputer
b. *Keyboard* dan *Mouse*
c. Monitor

5.3.4. Kebutuhan Antarmuka Perangkat Lunak

Perangkat-lunak yang dibutuhkan untuk mengoperasikan perangkat-lunak ini adalah:

a. Nama : Matlab 6.1
Sumber : The MathWorks, Inc.
Perangkat-lunak ini digunakan sebagi *tool* pembuatan aplikasi

b. Nama : Microsoft Windows 2000/ XP
Sumber : Microsoft
Perangkat lunak sebagai sistem operasi komputer

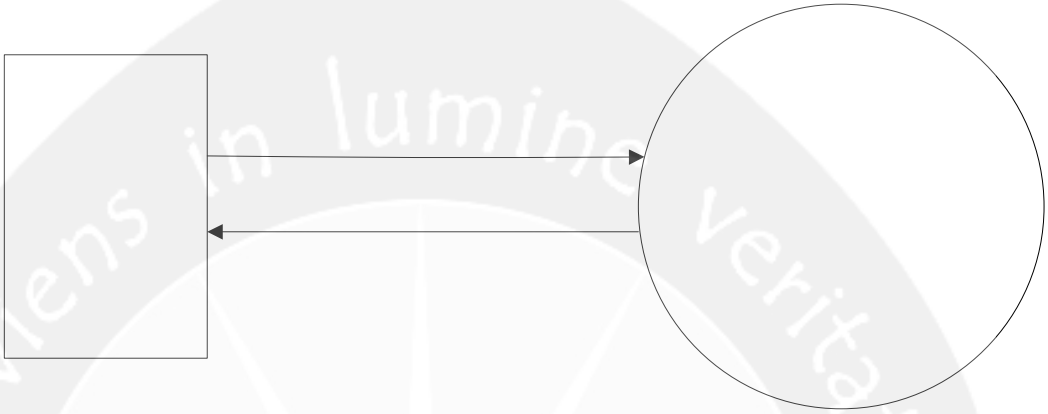
5.3.5. Kebutuhan Antarmuka Komunikasi

Dalam aplikasi ini tidak digunakan antarmuka komunikasi karena aplikasi berjalan secara *standalone*.

5.4. Kebutuhan Fungsionalitas

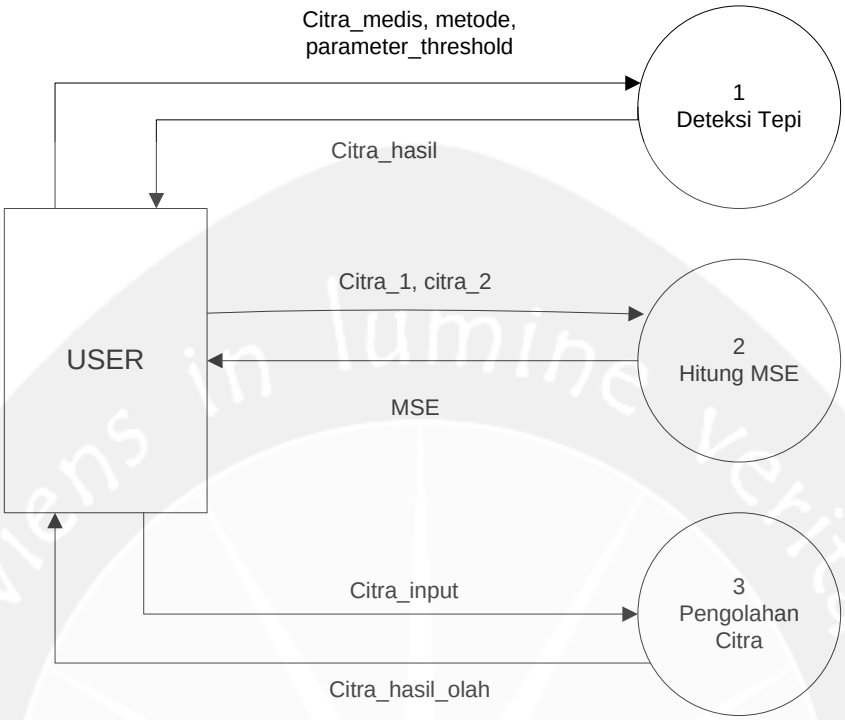
5.4.1. *Data Flow Diagram* (DFD)

Data Flow Diagram level 0 sistem ini dapat digambarkan sbb.



Gambar 5.2. DFD Level 0

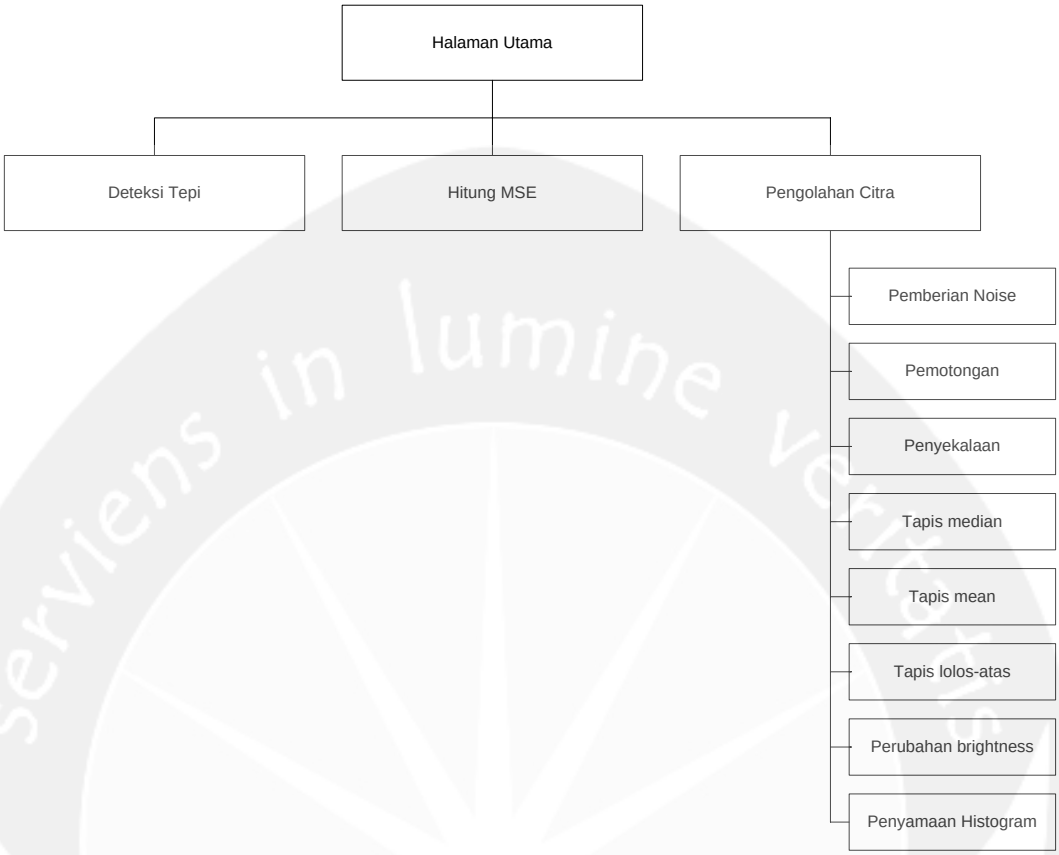
Sedangkan DFD level 1 dapat dilihat pada Gambar 5.3. Pada DFD level 1 terdapat tiga proses utama, yaitu: proses deteksi tepi, proses hitung MSE, dan proses pengolahan citra. Proses tersebut memerlukan parameter threshold yang diinputkan oleh user.



Gambar 5.3. DFD Level 1

5.4.2. Perancangan Arsitektur Modul

Berikut ini adalah gambar modul perancangan arsitektur sistem ini:



Gambar 5.4. Perancangan Arsitektural Modul

Gambar 5.4 menunjukkan bahwa program secara umum terdiri dari tiga bagian, yaitu: modul yang digunakan untuk deteksi tepi, modul yang digunakan untuk menghitung MSE, dan modul yang digunakan untuk pengolahan citra. Modul yang digunakan untuk pengolahan citra dibagi menjadi beberapa modul, yaitu: modul pemberian *noise*, pemotongan video, penyekalaan video, tapis median, tapis *mean*, tapis lolos-atas, perubahan *brightness*, dan penyamaan histogram.

BAB VI
HASIL DAN PEMBAHASAN

6.1 Hasil

Dalam pengujian program deteksi tepi citra medis dengan menggunakan algoritma kompas digunakan berkas citra dengan kedalaman piksel 24-bit warna dan berekstensi jpg dan bmp. Dalam penelitian ini digunakan beberapa citra uji seperti terlihat dalam Gambar 6.1 – 6.4.

Gambar 6.1. Citra elbow.jpg

Gambar 6.2. Citra hand.bmp

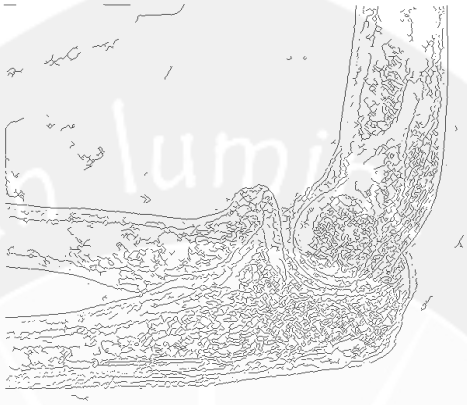
Gambar 6.3. Citra left-shoulder.bmp

Gambar 6.4. Citra lung.bmp

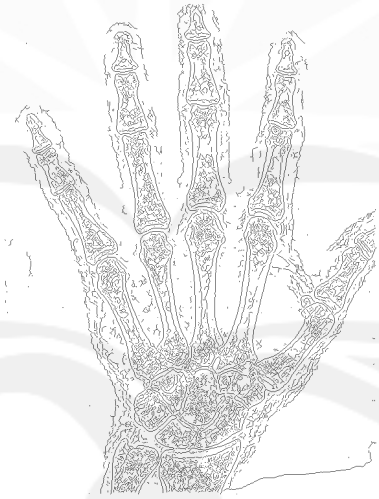
Pada penelitian ini akan dikembangkan algoritma deteksi tepi citra medis dengan algoritma Canny.

6.2. Deteksi Tepi Citra dengan menggunakan Detektor Canny

Gambar 6.5 – 6.8 merupakan hasil proses deteksi tepi citra dengan menggunakan detector Canny.

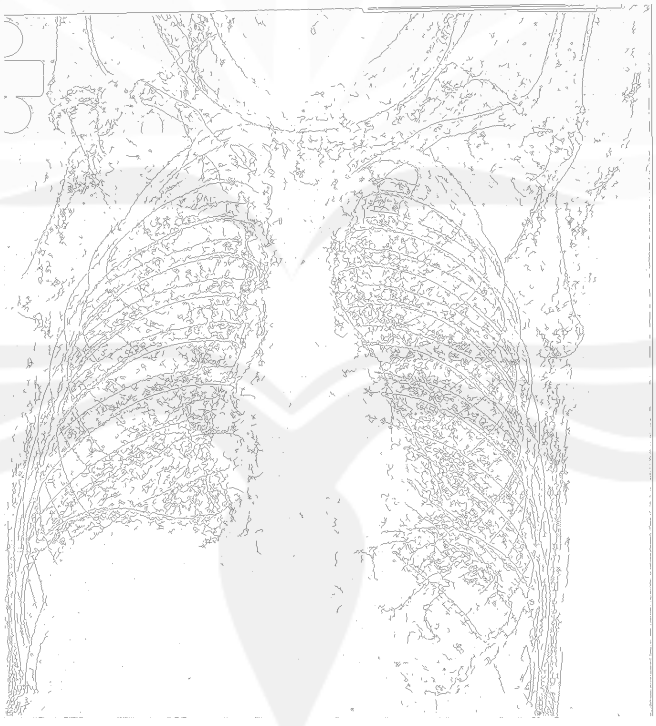


Gambar 6.5. Citra hasil deteksi tepi citra elbow.jpg menggunakan detector Canny



Gambar 6.6. Citra hasil deteksi tepi citra hand.jpg menggunakan Detektor Canny

Gambar 6.7. Citra hasil deteksi tepi citra left-shoulder.jpg detektor Canny



Gambar 6.8. Citra hasil deteksi tepi citra lung.jpg menggunakan detektor Canny

6.3. Pembahasan Program

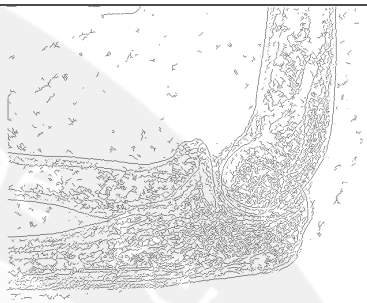
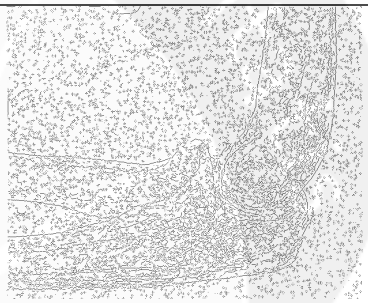
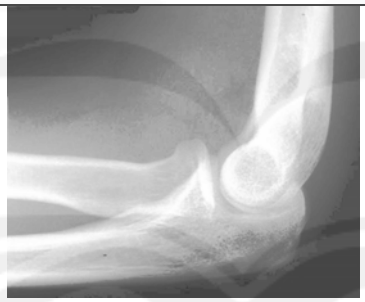
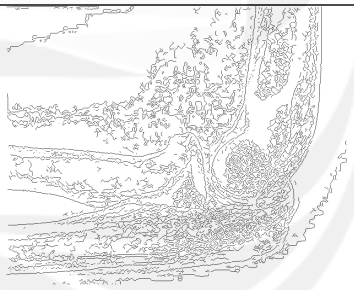
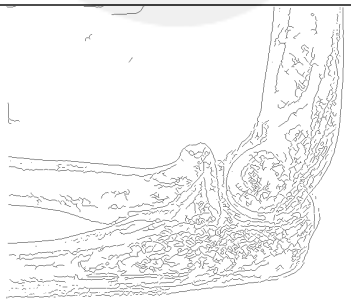
Cuplikan program dengan menggunakan Matlab berikut ini adalah digunakan untuk melakukan deteksi tepi citra dengan menggunakan detector Canny:

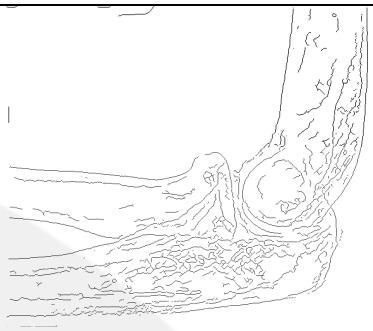
```
%deteksi dgn Canny
citra = imread('lung.jpg');
citra = rgb2gray(citra);
hasil = edge(citra,'canny');
imshow(not(hasil));
imwrite(not(hasil),'lungCanny.png', 'bitdepth', 1);
```

6.4. Pengaruh *Noise* dan Operasi Pengolahan Citra terhadap Algoritma yang Dikembangkan

Untuk uji pengaruh noise dan Operasi Pengolahan Citra akan digunakan citra elbow.jpg pada detector Canny. Tabel 6.1. merupakan hasil pengujian dengan menggunakan *noise Gaussian*, *noise salt and pepper*, *histogram equalization*, penapisan tapis lolos bawah (*Low Pass Filter*) dan penapisan tapis lolos atas (*High Pass Filter*).

Tabel 6.1. Hasil Ujicoba dengan *Noise* dan Operasi Pengolahan Citra Pada detector Canny

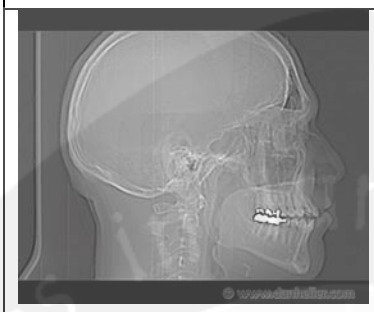
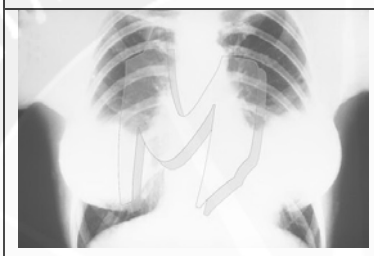
Gangguan	**Citra setelah Diolah**	**Hasil Deteksi Tepi**
Noise gaussian, noise varians = 0.0001		
Noise salt and pepper Intensitas 0.01		
Histogram Equalization		
Low Pass Filter 3x3		

Low Pass Filter 5x5		
High Pass Filter		

Berdasarkan hasil pengujian yang ditunjukkan pada Tabel 6.1, hasil deteksi tepi citra akan mengalami gangguan pada saat citra diberikan operasi pengolahan citra. Hasil operasi deteksi tepi citra yang dikembangkan akan mengalami gangguan yang signifikan apabila diberikan gangguan *noise salt and pepper*, *histogram equalization*, dan operasi penapisan dengan tapis lolos atas (*High Pass Filtering*). Ketiga operasi pengolahan citra ini akan menambahkan efek pseudo edge (tepi semu) pada hasil deteksi citranya.

Algoritma yang dikembangkan cukup dapat bertahan terhadap pengolahan citra pemberian *noise Gaussian* dan penapisan lolos bawah (*Low Pass Filtering*). Dengan menggunakan penapisan tapis lolos bawah akan diperoleh detail tepi yang lebih sedikit dibandingkan tanpa diolah dengan *Low Pass Filtering*. Pemberian *noise Gaussian* juga tidak mempunyai efek yang signifikan terhadap algoritma yang dikembangkan.

Tabel 6.3. Hasil Uji Coba dengan Citra Lain

Citra	Hasil Detektor Canny
	
	

Tabel 6.3 merupakan contoh hasil pengujian deteksi tepi dengan operator Canny. Berdasarkan penelitian yang dilakukan, parameter threshold sangat menentukan tepi yang diperoleh.

BAB VII
KESIMPULAN DAN SARAN

7.1. Kesimpulan

1. Aplikasi deteksi tepi citra medis menggunakan *canny detector* ini telah dapat dikembangkan
2. Hasil operasi deteksi tepi citra yang dikembangkan akan mengalami gangguan yang signifikan apabila diberikan gangguan *noise salt and pepper*, *histogram equalization,* dan operasi penapisan dengan tapis lolos atas (*High Pass Filtering*). Algoritma yang dikembangkan cukup dapat bertahan terhadap pengolahan citra pemberian *noise Gaussian* dan penapisan lolos bawah (*Low Pass Filtering*).

7.2. Saran

1. Dapat dikembangkan kernel matriks algoritma deteksi tepi citra agar sesuai dengan karakter citra medis.

DAFTAR PUSTAKA

Benjelloun Mohammed, Said Mahmoudi, 2007, *Mobility Estimation and Analysis in Medical X-ray Images Using Corners and Faces Contours Detection*, International Machine Vision and Image Processing Conference (IMVIP 2007), pp. 106-116

Bingrong Wu, Xie Mei, 2008, *An Interactive Segmentation of Medical Image Series*, 2008 International Seminar on Future BioMedical Information Engineering, pp. 7-10

Gonzalez Manuel Hidalgo, Arnau Mir Torres, Joan Torrens Sastre, 2009, *Noisy Image Edge Detection Using an Uninorm Fuzzy Morphological Gradient*, 2009 Ninth International Conference on Intelligent Systems Design and Applications, pp. 1335-1340

Lee Bin, Yan Jia-yong, Zhuang Tian-ge, 2001, *A Dynamic Programming Based Algorithm for Optimal Edge Detection in Medical Images*, International Workshop on Medical Imaging and Augmented Reality (MIAR '01)

Naef M., O. Kuebler, G. Szekely , R. Kikinis , M.E. Shenton, 1996, *Characterization and Recognition of 3D Organ Shape in Medical Image Analysis Using Skeletonization*, 1996 Workshop on Mathematical Methods in Biomedical Image Analysis (MMBIA '96), pp. 0139

Selvarasu N., Sangeetha Vivek, N.M. Nandhitha, 2007, *Performance Evaluation of Image Processing Algorithms for Automatic Detection and Quantification of Abnormality in Medical Thermograms*, International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007), pp. 388-393

Suzuki Kenji, Isao Horiba, Noboru Sugie, 2003, *Neural Edge Enhancer for Supervised Edge Enhancement from Noisy Images*, IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1582-1596

Thangam SV, Deepak, SK, 2009, *An Effective Edge Detection Methodology for Medical Images Based on Texture Discrimination*, Seventh International Conference on Advances in Pattern Recognition

Zeng Yanjun, Tu Chengyuan, Zhang Xiaojun, 2008, *Fuzzy-Set Based Fast Edge Detection of Medical Image*, 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery

LAMPIRAN M-FILE YANG DIGUNAKAN

```
function [eout,thresh] = edge(varargin)

[a,method,thresh,sigma,H,kx,ky] = parse_inputs(varargin{:});

% Transform to a double precision intensity image
if isa(a, 'uint8') | isa(a, 'uint16')
   a = im2double(a);
end

m = size(a,1);
n = size(a,2);
rr = 2:m-1; cc=2:n-1;

% The output edge map:
e = repmat(logical(uint8(0)), m, n);

if strcmp(method,'canny')
   % Magic numbers
   GaussianDieOff = .0001;
   PercentOfPixelsNotEdges = .7; % Used for selecting thresholds
   ThresholdRatio = .4;          % Low thresh is this fraction of the high.

   % Design the filters - a gaussian and its derivative

   pw = 1:30; % possible widths
   ssq = sigma*sigma;
   width = max(find(exp(-(pw.*pw)/(2*sigma*sigma))>GaussianDieOff));
   if isempty(width)
      width = 1;  % the user entered a really small sigma
   end
   t = (-width:width);
   len = 2*width+1;
   t3 = [t-.5; t; t+.5];            % We will average values at t-.5, t, t+.5
   gau = sum(exp(-(t3.*t3)/(2*ssq))).'/(6*pi*ssq);  % the gaussian 1-d filter
   dgau = (-t.* exp(-(t.*t)/(2*ssq))/ ssq).';       % derivative of a gaussian

   % Convolve the filters with the image in each direction
   % The canny edge detector first requires convolutions with
   % the gaussian, and then with the derivitave of a gauusian.
   % I convolve the filters first and then make a call to conv2
   % to do the convolution down each column.

   ra = size(a,1);
   ca = size(a,2);
```

```
ay = 255*a; ax = 255*a';

h = conv(gau,dgau);
ax = conv2(ax, h, 'same').';
ay = conv2(ay, h, 'same');
mag = sqrt((ax.*ax) + (ay.*ay));
magmax = max(mag(:));
if magmax>0
  mag = mag / magmax;   % normalize
end

% Select the thresholds
if isempty(thresh)
  [counts,x]=imhist(mag, 64);
  highThresh = min(find(cumsum(counts) > PercentOfPixelsNotEdges*m*n)) / 64;
  lowThresh = ThresholdRatio*highThresh;
  thresh = [lowThresh highThresh];
elseif length(thresh)==1
  highThresh = thresh;
  if thresh>=1
    error('The threshold must be less than 1.');
  end
  lowThresh = ThresholdRatio*thresh;
  thresh = [lowThresh highThresh];
elseif length(thresh)==2
  lowThresh = thresh(1);
  highThresh = thresh(2);
  if (lowThresh >= highThresh) | (highThresh >= 1)
    error('Thresh must be [low high], where low < high < 1.');
  end
end

% The next step is to do the non-maximum supression.
% We will accrue indices which specify ON pixels in strong edgemap
% The array e will become the weak edge map.
idxStrong = [];
for dir = 1:4
  idxLocalMax = cannyFindLocalMaxima(dir,ax,ay,mag);
  idxWeak = idxLocalMax(mag(idxLocalMax) > lowThresh);
  e(idxWeak)=1;
  idxStrong = [idxStrong; idxWeak(mag(idxWeak) > highThresh)];
end

rstrong = rem(idxStrong-1, m)+1;
cstrong = floor((idxStrong-1)/m)+1;
e = bwselect(e, cstrong, rstrong, 8);
```

```
  e = bwmorph(e, 'thin', 1);  % Thin double (or triple) pixel wide contours

elseif any(strcmp(method, {'log','marr-hildreth','zerocross'}))
  % We don't use image blocks here
  if isempty(H),
    fsize = ceil(sigma*3) * 2 + 1;  % choose an odd fsize > 6*sigma;
    op = fspecial('log',fsize,sigma);
  else
    op = H;
  end

  op = op - sum(op(:))/prod(size(op)); % make the op to sum to zero
  b = filter2(op,a);

  if isempty(thresh)
    thresh = .75*mean2(abs(b(rr,cc)));
  end

  % Look for the zero crossings:  +-, -+ and their transposes
  % We arbitrarily choose the edge to be the negative point
  [rx,cx] = find( b(rr,cc) < 0 & b(rr,cc+1) > 0 ...
    & abs( b(rr,cc)-b(rr,cc+1) ) > thresh );   % [- +]
  e((rx+1) + cx*m) = 1;
  [rx,cx] = find( b(rr,cc-1) > 0 & b(rr,cc) < 0 ...
    & abs( b(rr,cc-1)-b(rr,cc) ) > thresh );   % [+ -]
  e((rx+1) + cx*m) = 1;
  [rx,cx] = find( b(rr,cc) < 0 & b(rr+1,cc) > 0 ...
    & abs( b(rr,cc)-b(rr+1,cc) ) > thresh);   % [- +]'
  e((rx+1) + cx*m) = 1;
  [rx,cx] = find( b(rr-1,cc) > 0 & b(rr,cc) < 0 ...
    & abs( b(rr-1,cc)-b(rr,cc) ) > thresh);   % [+ -]'
  e((rx+1) + cx*m) = 1;

  % Most likely this covers all of the cases.   Just check to see if there
  % are any points where the LoG was precisely zero:
  [rz,cz] = find( b(rr,cc)==0 );
  if ~isempty(rz)
    % Look for the zero crossings: +0-, -0+ and their transposes
    % The edge lies on the Zero point
    zero = (rz+1) + cz*m;   % Linear index for zero points
    zz = find(b(zero-1) < 0 & b(zero+1) > 0 ...
      & abs( b(zero-1)-b(zero+1) ) > 2*thresh);     % [- 0 +]'
    e(zero(zz)) = 1;
    zz = find(b(zero-1) > 0 & b(zero+1) < 0 ...
      & abs( b(zero-1)-b(zero+1) ) > 2*thresh);     % [+ 0 -]'
    e(zero(zz)) = 1;
```

```
    zz = find(b(zero-m) < 0 & b(zero+m) > 0 ...
      & abs( b(zero-m)-b(zero+m) ) > 2*thresh);     % [- 0 +]
    e(zero(zz)) = 1;
    zz = find(b(zero-m) > 0 & b(zero+m) < 0 ...
      & abs( b(zero-m)-b(zero+m) ) > 2*thresh);     % [+ 0 -]
    e(zero(zz)) = 1;
  end

else  % one of the easy methods (roberts,sobel,prewitt)

  % Determine edges in blocks for easy methods
  nr = length(rr); nc = length(cc);

  blk = bestblk([nr nc]);
  nblks = floor([nr nc]./blk); nrem = [nr nc] - nblks.*blk;
  mblocks = nblks(1); nblocks = nblks(2);
  mb = blk(1); nb = blk(2);

  if strcmp(method,'sobel')
    op = [-1 -2 -1;0 0 0;1 2 1]/8; % Sobel approximation to derivative
    bx = abs(filter2(op',a)); by = abs(filter2(op,a));
    b = kx*bx.*bx + ky*by.*by;
    if isempty(thresh), % Determine cutoff based on RMS estimate of noise
      cutoff = 4*sum(sum(b(rr,cc)))/prod(size(b(rr,cc))); thresh = sqrt(cutoff);
    else                % Use relative tolerance specified by the user
      cutoff = (thresh).^2;
    end
    rows = 1:blk(1);
    for i=0:mblocks,
      if i==mblocks, rows = (1:nrem(1)); end
      for j=0:nblocks,
        if j==0, cols = 1:blk(2); elseif j==nblocks, cols=(1:nrem(2)); end
        if ~isempty(rows) & ~isempty(cols)
          r = rr(i*mb+rows); c = cc(j*nb+cols);
          e(r,c) = (b(r,c)>cutoff) & ...
          ( ( (bx(r,c) >= (kx*by(r,c)-eps*100)) & ...
          (b(r,c-1) <= b(r,c)) & (b(r,c) > b(r,c+1)) ) | ...
          ( (by(r,c) >= (ky*bx(r,c)-eps*100 )) & ...
          (b(r-1,c) <= b(r,c)) & (b(r,c) > b(r+1,c))));
        end
      end
    end

  elseif strcmp(method,'prewitt')
    op = [-1 -1 -1;0 0 0;1 1 1]/6; % Prewitt approximation to derivative
    bx = abs(filter2(op',a)); by = abs(filter2(op,a));
```

```
  b = kx*bx.*bx + ky*by.*by;
  if isempty(thresh), % Determine cutoff based on RMS estimate of noise
    cutoff = 4*sum(sum(b(rr,cc)))/prod(size(b(rr,cc))); thresh = sqrt(cutoff);
  else                % Use relative tolerance specified by the user
    cutoff = (thresh).^2;
  end
  rows = 1:blk(1);
  for i=0:mblocks,
    if i==mblocks, rows = (1:nrem(1)); end
    for j=0:nblocks,
      if j==0, cols = 1:blk(2); elseif j==nblocks, cols=(1:nrem(2)); end
      if ~isempty(rows) & ~isempty(cols)
        r = rr(i*mb+rows); c = cc(j*nb+cols);
        e(r,c) = (b(r,c)>cutoff) & ...
        ( ( (bx(r,c) >= (kx*by(r,c)-eps*100) ) & ...
        (b(r,c-1) <= b(r,c)) & (b(r,c) > b(r,c+1)) ) | ...
        ((by(r,c) >= (ky*bx(r,c)-eps*100) )  & ...
        (b(r-1,c) <= b(r,c)) & (b(r,c) > b(r+1,c)) ) );
      end
    end
  end

elseif strcmp(method, 'roberts')
  op = [1 0;0 -1]/sqrt(2); % Roberts approximation to diagonal derivative
  bx = abs(filter2(op,a)); by = abs(filter2(rot90(op),a));
  b = kx*bx.*bx + ky*by.*by;
  if isempty(thresh), % Determine cutoff based on RMS estimate of noise
    cutoff = 6*sum(sum(b(rr,cc)))/prod(size(b(rr,cc))); thresh = sqrt(cutoff);
  else                % Use relative tolerance specified by the user
    cutoff = (thresh).^2;
  end
  rows = 1:blk(1);
  for i=0:mblocks,
    if i==mblocks, rows = (1:nrem(1)); end
    for j=0:nblocks,
      if j==0, cols = 1:blk(2); elseif j==nblocks, cols=(1:nrem(2)); end
      if ~isempty(rows) & ~isempty(cols)
        r = rr(i*mb+rows); c = cc(j*nb+cols);
        e(r,c) = (b(r,c)>cutoff) & ...
        ( ( (bx(r,c) >= (kx*by(r,c)-eps*100)) & ...
        (b(r-1,c-1) <= b(r,c)) & (b(r,c) > b(r+1,c+1)) ) | ...
        ( (by(r,c) >= (ky*bx(r,c)-eps*100)) & ...
        (b(r-1,c+1) <= b(r,c)) & (b(r,c) > b(r+1,c-1)) ) );
      end
    end
  end
```

```
  else
    error([method,' is not a valid method.']);
  end
end

if nargout==0,
  imshow(e);
else
  eout = e;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Local Function : cannyFindLocalMaxima
%
function idxLocalMax = cannyFindLocalMaxima(direction,ix,iy,mag);
%
% This sub-function helps with the non-maximum supression in the Canny
% edge detector.  The input parameters are:
%
%   direction - the index of which direction the gradient is pointing,
%               read from the diagram below. direction is 1, 2, 3, or 4.
%   ix        - input image filtered by derivative of gaussian along x
%   iy        - input image filtered by derivative of gaussian along y
%   mag       - the gradient magnitude image
%
%    there are 4 cases:
%
%                         The X marks the pixel in question, and each
%         3     2         of the quadrants for the gradient vector
%       O----0----0       fall into two cases, divided by the 45
%     4 |        | 1      degree line.  In one case the gradient
%       |        |        vector is more horizontal, and in the other
%       O    X   O        it is more vertical.  There are eight
%       |        |        divisions, but for the non-maximum supression
%    (1)|        |(4)     we are only worried about 4 of them since we
%       O----O----O       use symmetric points about the center pixel.
%        (2)   (3)


[m,n,o] = size(mag);

% Find the indices of all points whose gradient (specified by the
% vector (ix,iy)) is going in the direction we're looking at.
```

```
switch direction
case 1
  idx = find((iy<=0 & ix>-iy)  | (iy>=0 & ix<-iy));
case 2
  idx = find((ix>0 & -iy>=ix)  | (ix<0 & -iy<=ix));
case 3
  idx = find((ix<=0 & ix>iy) | (ix>=0 & ix<iy));
case 4
  idx = find((iy<0 & ix<=iy) | (iy>0 & ix>=iy));
end

% Exclude the exterior pixels
if ~isempty(idx)
  v = mod(idx,m);
  extIdx = find(v==1 | v==0 | idx<=m | (idx>(n-1)*m));
  idx(extIdx) = [];
end

ixv = ix(idx);
iyv = iy(idx);
gradmag = mag(idx);

% Do the linear interpolations for the interior pixels
switch direction
case 1
  d = abs(iyv./ixv);
  gradmag1 = mag(idx+m).*(1-d) + mag(idx+m-1).*d;
  gradmag2 = mag(idx-m).*(1-d) + mag(idx-m+1).*d;
case 2
  d = abs(ixv./iyv);
  gradmag1 = mag(idx-1).*(1-d) + mag(idx+m-1).*d;
  gradmag2 = mag(idx+1).*(1-d) + mag(idx-m+1).*d;
case 3
  d = abs(ixv./iyv);
  gradmag1 = mag(idx-1).*(1-d) + mag(idx-m-1).*d;
  gradmag2 = mag(idx+1).*(1-d) + mag(idx+m+1).*d;
case 4
  d = abs(iyv./ixv);
  gradmag1 = mag(idx-m).*(1-d) + mag(idx-m-1).*d;
  gradmag2 = mag(idx+m).*(1-d) + mag(idx+m+1).*d;
end
idxLocalMax = idx(gradmag>=gradmag1 & gradmag>=gradmag2);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Local Function : parse_inputs
%
function [I,Method,Thresh,Sigma,H,kx,ky] = parse_inputs(varargin)
% OUTPUTS:
%   I      Image Data
%   Method Edge detection method
%   Thresh Threshold value
%   Sigma  standard deviation of Gaussian
%   H      Filter for Zero-crossing detection
%   kx,ky  From Directionality vector

error(nargchk(1,5,nargin));

I = varargin{1};

% Defaults
Method='sobel';
Thresh=[];
Direction='both';
Sigma=2;
H=[];
K=[1 1];

methods = {'canny','prewitt','sobel','marr-hildreth','log','roberts','zerocross'};
directions = {'both','horizontal','vertical'};

% Now parse the nargin-1 remaining input arguments

% First get the strings - we do this because the intepretation of the
% rest of the arguments will depend on the method.
nonstr = [];   % ordered indices of non-string arguments
for i = 2:nargin
  if ischar(varargin{i})
    str = lower(varargin{i});
    j = strmatch(str,methods);
    k = strmatch(str,directions);
    if ~isempty(j)
      Method = methods{j(1)};
      if strcmp(Method,'marr-hildreth')
        warning('"Marr-Hildreth" is an obsolete syntax, use "LoG" instead.');
      end
    elseif ~isempty(k)
      Direction = directions{k(1)};
```

```
    else
      error(['Invalid input string: ''' varargin{i} '''.']);
    end
  else
    nonstr = [nonstr i];
  end
end

% Now get the rest of the arguments

switch Method

case {'prewitt','sobel','roberts'}
  threshSpecified = 0;  % Threshold is not yet specified
  for i = nonstr
    if prod(size(varargin{i}))<=1 & ~threshSpecified % Scalar or empty
      Thresh = varargin{i};
      threshSpecified = 1;
    elseif prod(size(varargin{i}))==2  % The dreaded K vector
      warning(['BW = EDGE(... , K) is an obsolete syntax. '...
       'Use BW = EDGE(... , DIRECTION), where DIRECTION is a string.']);
      K=varargin{i};
    else
      error('Invalid input arguments');
    end
  end

case 'canny'
  Sigma = 1.0;          % Default Std dev of gaussian for canny
  threshSpecified = 0;  % Threshold is not yet specified
  for i = nonstr
    if prod(size(varargin{i}))==2 & ~threshSpecified
      Thresh = varargin{i};
      threshSpecified = 1;
    elseif prod(size(varargin{i}))==1
      if ~threshSpecified
        Thresh = varargin{i};
        threshSpecified = 1;
      else
        Sigma = varargin{i};
      end
    elseif isempty(varargin{i}) & ~threshSpecified
      % Thresh = [];
      threshSpecified = 1;
    else
      error('Invalid input arguments');
```

```
    end
  end

case 'log'
  threshSpecified = 0;  % Threshold is not yet specified
  for i = nonstr
    if prod(size(varargin{i}))<=1  % Scalar or empty
      if ~threshSpecified
        Thresh = varargin{i};
        threshSpecified = 1;
      else
        Sigma = varargin{i};
      end
    else
      error('Invalid input arguments');
    end
  end

case 'zerocross'
  threshSpecified = 0;  % Threshold is not yet specified
  for i = nonstr
    if prod(size(varargin{i}))<=1 & ~threshSpecified % Scalar or empty
      Thresh = varargin{i};
      threshSpecified = 1;
    elseif prod(size(varargin{i})) > 1 % The filter for zerocross
      H = varargin{i};
    else
      error('Invalid input arguments');
    end
  end

case 'marr-hildreth'
  for i = nonstr
    if prod(size(varargin{i}))<=1  % Scalar or empty
      Thresh = varargin{i};
    elseif prod(size(varargin{i}))==2  % The dreaded K vector
      warning('The [kx ky] direction factor has no effect for "Marr-Hildreth".');
    elseif prod(size(varargin{i})) > 2 % The filter for zerocross
      H = varargin{i};
    else
      error('Invalid input arguments');
    end
  end

otherwise
  error('Invalid input arguments');
```

```
end

if Sigma<=0
  error('Sigma must be positive');
end

switch Direction
case 'both',
  kx = K(1); ky = K(2);
case 'horizontal',
  kx = 0; ky = 1; % Directionality factor
case 'vertical',
  kx = 1; ky = 0; % Directionality factor
otherwise
  error('Unrecognized direction string');
end


if isrgb(I)
  error('RGB images are not supported. Call RGB2GRAY first.');
end
```